Rapid Monte Carlo Simulation for Forecasting, Stress Testing, and Scenario Analysis

Parallel Processing in Apache Spark

May 17, 2016
 Agenda

1. Introductions
2. Monte Carlo Examples
3. Code Improvement
4. Vasicek Analysis
5. Apache Spark Implementation
6. Q&A
Introductions

- FI Consulting (FI) specializes in providing financial institution clients with custom analytics, model validation and advisory services, and development support for financial and analytic software. FI focuses on diverse sectors of the credit market including mortgages, consumer finance, and small business loans.

- Maxwell Consulting provides deep expertise in the most complicated sectors of the financial world, whether analyzing the valuation of a derivative transaction or advising on the quality of a firm’s risk management. Maxwell serves diverse audiences including executives, legal counsel, traders, auditors, and jurors.
Poll Questions
Monte Carlo Advice

1. Monte Carlo Advice
2. Introductions
3. Monte Carlo Examples
4. Code Improvement
5. Vasicek Analysis
6. Apache Spark Implementation
7. Q&A
Monte Carlo Advice

Value of MC Simulation increases when calculations run faster

More precision in results

Permits greater “realism” in adding features

Run more “experiments”

Examples: Real-time risk management; nested MC methods
Monte Carlo Advice (cont.)

Almost always possible to get faster speed

- Hardware – e.g., high-performance processors in parallel
- Software – e.g., optimized code

Disadvantage of “off-the-shelf” MC products
Monte Carlo Examples

1. Introductions
2. Monte Carlo Advice
3. Examples
4. Code Improvement
5. Vasicek Analysis
6. Apache Spark Implementation
7. Q&A
Monte Carlo Example #1: Stochastic Models

Yield $y$ with Standard Deviation $s$, with Random Variable $\varepsilon \rightarrow N(0,1)$

- **“Mean Reversion” SDE:**
  \[ \Delta y_i = \omega (m - y_i) \Delta t + s \varepsilon \sqrt{\Delta t} \]

- **“Normal” SDE:**
  \[ \Delta y_i = r \Delta t + s \varepsilon \sqrt{\Delta t} \]

- **“Log-Normal” SDE:**
  \[ \Delta y_i = \mu y_i \Delta t + \sigma y_i \varepsilon \sqrt{\Delta t} \]
Monte Carlo Example #2: Simulation for Collateral Defaults

Set Initial Conditions

- Assets
- Liabilities
- Terms
Monte Carlo Example #2: Simulation for Collateral Defaults

Set Initial Conditions
- Assets
- Liabilities
- Terms

Take a Step Forward in Time
- Change yield curve and other market variables
- Check for asset defaults
- Record scheduled asset payments
- Make scheduled liability payments
Monte Carlo Example #2: Simulation for Collateral Defaults

Set Initial Conditions
- Assets
- Liabilities
- Terms

Take a Step Forward in Time
- Change yield curve and other market variables
- Check for asset defaults
- Record scheduled asset payments
- Make scheduled liability payments

Record the Loss to Investors (if any)
- Return to Step 2 if haven’t reached maturity
Monte Carlo Example #2: Simulation for Collateral Defaults

Set Initial Conditions
- Assets
- Liabilities
- Terms

Take a Step Forward in Time
- Change yield curve and other market variables
- Check for asset defaults
- Record scheduled asset payments
- Make scheduled liability payments

Record the Loss to Investors (if any)
- Return to Step 2 if haven’t reached maturity

Estimate Density Functions for Losses to Each Tranche
Monte Carlo Example #3: Simulation for Bank Asset Defaults

Banking Simulator at Bankmaxwell.azurewebsites.net

©2016 FI Consulting. All rights reserved.
Monte Carlo Example #4: Vasicek Default Model

- Consider Portfolio of “Infinite” Number of Loans
- Each Loan has Identical Default Probability
- Each Loan has Identical Correlation to all Other Loans
- Vasicek: Analytical Solution for Portfolio Loss Distribution

\[ F(x) = \Phi \left[ -\frac{K + \sqrt{1-\rho} \Phi^{-1}(x)}{\sqrt{\rho}} \right] \quad \text{with} \quad K = \Phi^{-1}(p) \]
Vasicek Analysis
Vasicek Analysis

- Real Project: Analysis of Auto Loan Pool Losses
- With Many Pools, MLE Estimation of PD & Correlation
- Need a Test for the MLE Estimators
- Build Monte Carlo Algorithm

\[ F(x) = \Phi \left[ \frac{-K + \sqrt{1 - \rho \Phi^{-1}(x)}}{\sqrt{\rho}} \right] \quad \text{with} \quad K \equiv \Phi^{-1}(p) \]
Vasicek Analysis

- Imagine M Loans in Each of the N Pools
- Monte Carlo Simulation to get Default Fraction each Pool
- Check MLE Estimators given “known” PD & Correlation
- Also a Test of Vasicek and Numerical Methods
Vasicek Analysis

- The **Vasicek distribution** describes the probability density function for the fraction of defaulted loans within an infinitely diversified portfolio. Simple but restrictive assumptions specify a single default probability (PD) common to each loan and a single correlation parameter linking the behavior of all loans.

- **Variance test**: the numerical integration and Monte Carlo simulation are two viable methods to compute the variance of this Vasicek distribution - very important for understanding the risk of the loan portfolio!

- **Pool Test**: Create an arbitrary number of pools with an arbitrary number of obligors per pool. Apply Monte Carlo simulation to determine fraction of defaults in each pool. Then compare to the analytical Vasicek distribution.
Vasicek Analysis

Visual Studio .Net Web Application at Webjoe.azurewebsites.net/Vasicek_Variance
Code Improvement

1. Introductions
2. Monte Carlo Examples
3. Code Improvement
4. Monte Carlo Advice
5. Vasicek Analysis
6. Apache Spark Implementation
7. Q&A
Code Improvement

' Generate Num_Pools to measure the default fraction of Number_Obligors within each pool. We provide a single-factor correlation and then determine if the MLE extraction for correlation and Obligor default probability works well.
For kount_Pools = 1 To Num_pools
  Def_number = 0
  ' Set the systemic random variable Pool_Y.
  Call Gauss_RV(G1, G2)
  Pool Y = G1 * Sqr(rho)
  For kount = 1 To Number_obligors Step 2
    Call Gauss_RV(G1, G2)
    Ob_RV = Pool_Y + Sqr(1# - rho) * G1
    If Application.NormSDist(Ob_RV) < Def_prob Then Def_number = Def_number + 1
    Ob_RV = Pool_Y + Sqr(1# - rho) * G2
    If Application.NormSDist(Ob_RV) < Def_prob Then Def_number = Def_number + 1
  Next kount
  Def_fraction(kount_Pools) = CDb1(Def_number) / CDb1(Number_obligors)
  If Def_number = 0 Then
    x(kount_Pools) = Min_X
  Else
    x(kount_Pools) = Application.NormSInv(Def_fraction(kount_Pools))
  End If
Next kount_Pools
Code Improvement

' Generate Num_Pools to measure the default fraction of Number_Obligors within each pool. We provide a single-factor correlation and then determine if the MLE extraction for correlation and Obligor default probability works well.

For kount_Pools = 1 To Num_pools
    Def_number = 0
    ' Set the systemic random variable Pool_Y.
    Call Gauss_RV(G1, G2)
    Pool_Y = G1 * Sqr(rho)
    For kount = 1 To Number_obligors Step 2
        Call Gauss_RV(G1, G2)
        Ob_RV = Pool_Y + Sqr(1# - rho) * G1
        If Application.NormSDist(Ob_RV) < Def_prob Then Def_number = Def_number + 1
        Ob_RV = Pool_Y + Sqr(1# - rho) * G2
        If Application.NormSDist(Ob_RV) < Def_prob Then Def_number = Def_number + 1
    Next kount

Def_fraction(kount_Pools) = CDb1(Def_number) / CDb1(Number_obligors)
If Def_number = 0 Then
    x(kount_Pools) = Min_X
Else
    x(kount_Pools) = Application.NormSInv(Def_fraction(kount_Pools))
End If

Next kount_Pools

With 2,000 Pools and 10,000 Obligors per Pool, the Inner Loop Generates RVs for Default Determination of 20 Million Loans
Code Improvement

' Generate Num_Pools to measure the default fraction of Number_Obligors
' within each pool. We provide a single-factor correlation and then
' determine if the MLE extraction for correlation and Obligor default
' probability works well.
For kount_Pools = 1 To Num_pools
    Def_number = 0
    ' Set the systemic random variable Pool_Y.
    Call Gauss_RV(G1, G2)
    Pool Y = G1 * Sqr(rho)
For kount = 1 To Number_obligors Step 2
    Call Gauss_RV(G1, G2)
    Ob_RV = Pool_Y + Sqr(1 - rho) * G1
    If Application.NormSDist(Ob_RV) < Def_prob Then Def_number = Def_number + 1
    Ob_RV = Pool_Y + Sqr(1 - rho) * G2
    If Application.NormSDist(Ob_RV) < Def_prob Then Def_number = Def_number + 1
Next kount
Def_fraction(kount_Pools) = CDBl(Def_number) / CDBl(Number_obligors)
If Def_number = 0 Then
    x(kount_Pools) = Min_X
Else
    x(kount_Pools) = Application.NormSInv(Def_fraction(kount_Pools))
End If
Next kount Pools

Look Carefully at Each Line of Code in the Inner Loop to Reduce “Expensive” Calculations – ex: redundant Sqr evaluations
Look Carefully at Each Line of Code in the Inner Loop to Reduce “Expensive” Calculations – ex: Normal CDF determinations
Code Improvement

Generate Num_Pools to measure the default fraction of Number_Obligors within each pool. We provide a single-factor correlation and then determine if the MLE extraction for correlation and Obligor default probability works well.

For kount_Pools = 1 To Num_pools
  Def_number = 0
  ' Set the systemic random variable Pool_Y.
  Call Gauss RV(G1, G2)
  Pool Y = G1 * Sqr(rho)

  For kount = 1 To Number_obligors Step 2
    Call Gauss RV(G1, G2)
    Ob_RV = Pool_Y + Sqr(1# - rho) * G1
    If Application.NormSDist(Ob_RV) < Def_prob Then Def_number = Def_number + 1
    Ob_RV = Pool_Y + Sqr(1# - rho) * G2
    If Application.NormSDist(Ob_RV) < Def_prob Then Def_number = Def_number + 1
  Next kount

  Def_fraction(kount_Pools) = CDb1(Def_number) / CDb1(Number_obligors)
  If Def_number = 0 Then
    x(kount_Pools) = Min_X
  Else
    x(kount_Pools) = Application.NormSInv(Def_fraction(kount_Pools))
  End If

Next kount_Pools
Code Improvement

\[
\begin{align*}
\text{Sq}_\text{rho} &= \text{Sqr} (\text{rho}) \\
\text{Sq}_\text{wmr}_\text{rho} &= \text{Sqr} (1 - \text{rho}) \\
\end{align*}
\]

' Generate Num_Pools to measure the default fraction of Number_Obligors
' within each pool. We provide a single-factor correlation and then
' determine if the MLE extraction for correlation and Obligor default
' probability works well.
For kount_Pools = 1 To Num_pools
    Def_number = 0
    ' Set the systemic random variable Pool_Y.
    \text{Call Gauss_RV} (G1, G2)
    Pool_Y = G1 \ast \text{Sq}_\text{rho}
    For kount = 1 To Number_obligors Step 2
        \text{Call Gauss_RV} (G1, G2)
        Ob_RV = Pool_Y + \text{Sq}_\text{wmr}_\text{rho} \ast G1
        If Application.NormSDist(Ob_RV) < Def_prob Then Def_number = Def_number + 1
        Ob_RV = Pool_Y + \text{Sq}_\text{wmr}_\text{rho} \ast G2
        If Application.NormSDist(Ob_RV) < Def_prob Then Def_number = Def_number + 1
    Next kount
    Def_fraction(kount_Pools) = \text{CDBl} (\text{Def_number}) / \text{CDBl} (\text{Number_obligors})
    If Def_number = 0 Then
        x(kount_Pools) = \text{Min}_X
    Else
        x(kount_Pools) = Application.NormSInv(Def_fraction(kount_Pools))
    End If
Next kount_Pools
Code Improvement

Sq_rho = Sqr(rho)
Sq_wmrho = Sqr(1# - rho)

' Generate Num_Pools to measure the default fraction of Number_Obligors
' within each pool. We provide a single-factor correlation and then
' determine if the MLE extraction for correlation and Obligor default
' probability works well.
For kount_Pools = 1 To Num_pools
  Def_number = 0
  ' Set the systemic random variable Pool_Y.
  Call Gauss_RV(G1, G2)
  Pool_Y = G1 * Sq_rho
  Call Gauss_RV(G1, G2)
  Ob_RV = Pool_Y + Sq_wmrho * G1
  If JNormsDist(Ob_RV) < Def_prob Then Def_number = Def_number + 1
  Ob_RV = Pool_Y + Sq_wmrho * G2
  If JNormsDist(Ob_RV) < Def_prob Then Def_number = Def_number + 1
  Next kount

Def_fraction(kount_Pools) = CDbl(Def_number) / CDbl(Number_Obligors)
If Def_number = 0 Then
  x(kount_Pools) = Min X
Else
  x(kount_Pools) = Application.NormSInv(Def_fraction(kount_Pools))
End If
Next kount_Pools

Change: Reduces Execution Time from 117 to 28 – Big Saving!
Code Improvement

\[ S_q \_r h o = S q r (r h o) \]
\[ S_q \_w m r h o = S q r (1 - r h o) \]
\[ \Phi \_i n v \_D e f \_p r o b = A p p l i c a t i o n . N o r m S I n v (D e f \_p r o b) \]

' Generate Num_Pools to measure the default fraction of Number_Obligors' ' within each pool. We provide a single-factor correlation and then' ' determine if the MLE extraction for correlation and Obligor default' ' probability works well.

For kount_Pools = 1 To Num_pools
  Def_number = 0
  ' Set the systemic random variable Pool_Y.
  Call Gauss_RV(G1, G2)
  Pool_Y = G1 * S_q_rho
  For kount = 1 To Number_obligors Step 2
    Call Gauss_RV(G1, G2)
    Ob_RV = Pool_Y + S_q_wmrho * G1
    If Ob_RV < Phi_inv_Def_prob Then Def_number = Def_number + 1
    Ob_RV = Pool_Y + S_q_wmrho * G2
    If Ob_RV < Phi_inv_Def_prob Then Def_number = Def_number + 1
  Next kount
  Def_fraction(kount_Pools) = CDb1(Def_number) / CDb1(Number_obligors)
  If Def_number = 0 Then
    x(kount_Pools) = Min_X
  Else
    x(kount_Pools) = Application.NormSInv(Def_fraction(kount_Pools))
  End If
Next kount_Pools
Code Improvement

Sg_rho = Sqr(rho)
Sg_wmrho = Sqr(1# - rho)
Phi_inv_Def_prob = Application.NormSInv(Def_prob)

' Generate Num_Pools to measure the default fraction of Number_Obligors within each pool. We provide a single-factor correlation and then determine if the MLE extraction for correlation and Obligor default probability works well.
For kount_Pools = 1 To Num_pools
  Def_number = 0
  ' Set the systemic random variable Pool_Y.
  Call Gauss_RV(G1, G2)
  Pool_Y = G1 * Sq_rho
  For kount = 1 To Number_obligors Step 2
    Call Gauss_RV(G1, G2)
    Ob_RV = Pool_Y + Sg_wmrho * G1
    If Ob_RV < Phi_inv_Def_prob Then Def_number = Def_number + 1
    Ob_RV = Pool_Y + Sg_wmrho * G2
    If Ob_RV < Phi_inv_Def_prob Then Def_number = Def_number + 1
  Next kount
Def_fraction(kount_Pools) = CDbl(Def_number) / CDbl(Number_obligors)
If Def_number = 0 Then
  x(kount_Pools) = Min_X
Else
  x(kount_Pools) = Application.NormSInv(Def_fraction(kount_Pools))
End If
Next kount_Pools

Change: Reduces Execution Time from 28 to 8.4 – Big Saving!
Code Improvement

' Box-Muller generation of independent Gaussian random variates
' (mean zero and variance one) from Uniform independent random variates
Sub Gauss_RV(G1 As Double, G2 As Double)
  Dim U1, U2 As Double
  U1 = Rnd(1)
  U2 = Rnd(1)
  G1 = Sqr(-2 * Log(U1)) * Cos(Two_Pi * U2)
  G2 = Sqr(-2 * Log(U1)) * Sin(Two_Pi * U2)
End Sub

- Change: “Fixing” the Random Value Routine has Little Benefit
- End Result is 15x Speed Improvement by Altering the Monte Carlo Inner Loop
Apache Spark Implementation
Serial vs. Parallel Programming

- A Serial Program consists of a sequence of instructions, where each instruction executes one after the other.

- In a Parallel Program, the processing is broken up into parts, each of which could be executed concurrently on a different processor.
What is MapReduce?

- A **map job**, takes a set of data and converts it into another set of data, where individual elements are broken down into key/value pairs.

- A **reduce job** takes the output from a map as input and **merges together** these values to form a possibly smaller set of values.
Spark

- Spark is an open-source software solution that performs rapid calculations on in-memory distributed datasets.

- Spark’s uses Resilient Distributed Datasets (RDDs). RDDs can be automatically recomputed on failure and are resilient and fault-tolerant.
Parallel Processing of Monte Carlo Samples

```python
def default(idx):
    def_number = 0
    temp_gauss = Gauss_RV()
    ob_rv = pool_y + sq_wmrho * temp_gauss[0]
    if ob_rv < phi_inv_def_prob: def_number += 1
    ob_rv = pool_y + sq_wmrho * temp_gauss[1]
    if ob_rv < phi_inv_def_prob: def_number += 1
    return def_number

for kount_pools in range(0,num_pools):
    temp_gauss = Gauss_RV()
    pool_y = temp_gauss[0] * sq_rh0
    count = sc.parallelize(xrange(number_obligors_half)).map(default)
    count = count.reduce(add)

    def_fraction[kount_pools] = count / float(number_obligors)
```
Parallel Processing of Monte Carlo Samples

```python
def default(idx):
    def_number = 0
    temp_gauss = Gauss_RV()
    ob rv = pool y + sq wmrho * temp_gauss[0]
    if ob rv < phi_inv_def_prob: def_number += 1
    ob rv = pool y + sq wmrho * temp_gauss[1]
    if ob rv < phi_inv_def_prob: def_number += 1
    return def_number

for kount_pools in range(0,num_pools):
    temp_gauss = Gauss_RV()
    pool_y = temp_gauss[0] * sq_rho
    count = sc.parallelize(xrange(number_obligors_half)).map(default)
    count = count.reduce(add)
    def_fraction[kount_pools] = count / float(number_obligors)
```
Parallel Processing of Monte Carlo Samples

```python
def default(idx):
    def_number = 0
    temp_gauss = Gauss_RV()
    ob_rv = pool_y + sq_wmrho * temp_gauss[0]
    if ob_rv < phi_inv_def_prob: def_number += 1
    ob_rv = pool_y + sq_wmrho * temp_gauss[1]
    if ob_rv < phi_inv_def_prob: def_number += 1
    return def_number

for kount_pools in range(0,num_pools):
    temp_gauss = Gauss_RV()
    pool_y = temp_gauss[0] * sq_rho

    count = sc.parallelize(xrange(number_obligors_half)).map(default)
    count = count.reduce(add)

    def_fraction[kount_pools] = count / float(number_obligors)
```
def Gauss_RV():
    u1 = random.random()
    u2 = random.random()
    u3 = random.random()
    u4 = random.random()
    g1 = sqrt(-2*math.log(u1)) * math.cos(two_pi*u2)
    g2 = sqrt(-2*math.log(u1)) * math.sin(two_pi*u2)
    g3 = sqrt(-2*math.log(u3)) * math.cos(two_pi*u4)
    g4 = sqrt(-2*math.log(u3)) * math.sin(two_pi*u4)
    gcorr1 = (LGDrho * g1) + (LGDrhocalc * g3)
    gcorr2 = (LGDrho * g2) + (LGDrhocalc * g4)
    return g1, g2, gcorr1, gcorr2

def default(idx):
    def_number = 0
    temp_gauss = Gauss_RV()
    ob_rv = pool_y + sq_wmrho * temp_gauss[0]
    if ob_rv < phi_inv_def_prob: def_number += (1-norm(0,1).cdf(temp_gauss[2]))
    ob_rv = pool_y + sq_wmrho * temp_gauss[1]
    if ob_rv < phi_inv_def_prob: def_number += (1-norm(0,1).cdf(temp_gauss[3]))
    return def_number
def Gauss_RV():
    u1 = random.random()
    u2 = random.random()
    u3 = random.random()
    u4 = random.random()
    g1 = sqrt(-2*math.log(u1)) * math.cos(two_pi*u2)
    g2 = sqrt(-2*math.log(u1)) * math.sin(two_pi*u2)
    g3 = sqrt(-2*math.log(u3)) * math.cos(two_pi*u4)
    g4 = sqrt(-2*math.log(u3)) * math.sin(two_pi*u4)
    gcorr1 = (LGDrho * g1) + (LGDrhocalc * g3)
    gcorr2 = (LGDrho * g2) + (LGDrhocalc * g4)
    return g1, g2, gcorr1, gcorr2

def default(idx):
    def_number = 0
    temp_gauss = Gauss_RV()
    ob rv = pool y + sq wmrho * temp_gauss[0]
    if ob rv < phi_inv_def_prob: def_number += (1-norm(0,1).cdf(temp_gauss[2]))
    ob rv = pool y + sq wmrho * temp_gauss[1]
    if ob rv < phi_inv_def_prob: def_number += (1-norm(0,1).cdf(temp_gauss[3]))
    return def_number
Running Monte Carlo in the Cloud

Cluster: My cluster

Cluster ready to run steps.

Connections:
Enable Web Connection – Spark History Server, Ganglia, Resource Manager ...
(View All)

Master public DNS:
ec2-compute-1.amazonaws.com

SSH

Tags:
-- View All / Edit

Summary
ID: j-
Creation date: 2016-05-13 08:55 (UTC-4)
Elapsed time: 35 minutes
Auto-terminate: No
Termination protection: Off Change

Configuration Details
Release label: emr-4.6.0
Hadoop Amazon 2.7.2
distribution:
Applications: Ganglia 3.7.2, Spark 1.6.1
Log URI:
EMRFS Disabled
consistent
view:

Network and Hardware
Availability us-east-1d
zone:
Subnet ID: subnet-
Master: Running 1 m3.xlarge
Core: Running 2 m3.xlarge
Task: --
Running Monte Carlo in the Cloud (cont.)

```
[hadoop@ip-]
```
Running Monte Carlo in the Cloud (cont.)
Wrap-Up and Q&A!
Wrap-Up and Q&A!

To learn more, please contact us!

Robert Chang, Model Validation Lead
chang@ficonsulting.com
FI Consulting at www.ficonsulting.com

Joe Pimbley, Principal
pimbley@maxwell-consulting.com
Maxwell Consulting at www.maxwell-consulting.com